Integrating Z Into Large Projects
Tools and Techniques

Anthony Hall

Overview

This paper addresses the problem of using Z to specify large software intensive systems. It is addressed particularly at safety, security or business-critical systems that require a rich language like Z to specify complex data and behaviour. Examples of such systems include air traffic control [1] and financial systems [2]. I am particularly concerned here with the process of writing a system specification.

I have claimed [3] that this process is beneficial in itself regardless of any subsequent analysis or proof. The benefits are not automatic, however. The specification is only useful if

1. it is written as part of the requirements elicitation and analysis process;
2. it is comprehensible to all the stakeholders;
3. it is grounded in the application domain;
4. it expresses all the concepts that are relevant;
5. it is integrated into the rest of the development and verification process.

On a purely practical level, this implies that we want Z to be part of the ordinary documents that are used every day on the project. That means, in practice, that it has to be integrated into Microsoft Word. So I first describe a tool for writing and checking Z within the Word environment. That is only the start, however: we also need a process for writing the specification and guidelines for its structure. I will describe some progress we have made towards these goals.

A Z Tool for Microsoft Word

The de facto standard for writing Z is the LaTeX mark up first introduced by Spivey and now incorporated into the Z standard. Industry, however, does not use LaTeX: it uses Microsoft Word. I have developed, with help from several colleagues, a package of Z tools for Word. This is in day to day use on a large project and is being made freely available at http://sourceforge.net/projects/zwordtools.

The tool includes:

1. styles for laying out schemas and other Z paragraphs;
2. a Unicode font that includes all the Z symbols and is visually compatible with Times New Roman
3. automatic layout of Z paragraphs like the LaTeX equivalent, with italic text, formatted keywords and so on: for example

```
_BirthdayBook______________________________
| known : ℙ Name
| birthday : Name ⇸ Date
|______________________
| known = dom birthday
|__________________________________________
```

4. the ability to enter symbols from a palette or (for died in the wool LaTeX hackers) typing in the markup;
5. one-click typechecking, with errors highlighted in the Word document;
6. generation of indexes and cross-references to definition and use of Z names;
7. the ability to hide the Z completely so the document can be used by maths-phobic readers;
8. miscellaneous tools such as checking matching brackets.

The intention of the tool is

- to lower the barrier to the uptake of Z by removing at least one obstacle, the need to learn another document production method;
- to allow easy integration of Z with natural language, diagrams, tables and other notations relevant to the domain;
- to encourage incremental development of Z specifications by allowing frequent typechecking;
- to encourage the writing of good natural language by producing documents with the mathematics hidden.

The tool currently uses Mike Spivey's fuzz as its underlying typechecking engine. It works by exporting LaTeX mark up and importing the fuzz error report: the intention is that this mechanism could be used with other tools, in particular tools supporting the Z standard. Typechecking does not require declaration before use and works across multiple documents: for example an operations document can be checked against a separate data model document.

Writing Literate Z

It is crucial to realise that in a Z specification, the mathematics is subsidiary to the natural language. A piece of mathematics makes no sense unless we know the intended meaning of each construct. We therefore enforce a rule that an English description must precede the corresponding Z. The English and the Z are complementary: the English describes the relevant real-world concept and explains the meaning of every term in the maths; the maths makes precise the relationships between the terms defined in the English. Note in particular that there is no rule like "In case of a discrepancy between English and Z, the Z takes precedence": rather, the rule is that such a discrepancy is an error which must be corrected.

Data Model

Every schema that defines part of the system state is preceded by an explanation of what properties of the real world are being described. Here is a small example:

Attribute Name	**Definition**
position	The x and y co-ordinates in system coordinates
smoothedAltitude	The uncorrected altitude from the mode C in feet.

```
─Track──────────────────────────────────
│ Position
│ smoothedAltitude : optional Altitude
└───────────────────────────────────────
```

Altitude is a natural number that represents uncorrected height above mean sea level in feet. It is the height that an altimeter calibrated to a pressure of 1013.2 millibars at sea level would read. Its correspondence with physical height depends on the current pressure.

$$Altitude == \mathbb{N}$$

Operations

Here is an example of an operation description in the same style. Again the English stands alone and explains the real world meaning of the operation.

Issuing a clearance updates the known clearances.
Inputs are

controller?:	the controller who is issuing the clearance
flight?:	the flight for which the clearance is being issued
clearance?	the new clearance.

The operation is only allowed if the conditions described in *IssueClearanceAvailable* are met. In that case, the clearance is recorded as issued by the controller.

$$\begin{array}{|l}
\multicolumn{1}{l}{\underline{\;IssueClearance\;}}\\
\Delta\ Clearances\\
controller? : CONTROLLER\\
flight? : FLIGHT\\
clearance? : CLEARANCE\\
\hline
IssueClearanceAvailable \Rightarrow clearanceIssuedBy' =\\
\qquad\qquad clearanceIssuedBy \cup \{clearance? \mapsto controller?\}\\
\hline
\end{array}$$

ⓘ The new clearance is guaranteed to be unique so *clearanceIssuedBy* is guaranteed to remain functional under this update.

The last paragraph is an example of a *Z Comment*. This is not to explain the real world meaning, but rather to explain technicalities of the Z. If you hide the mathematics the Z comment is also hidden, since it is of no interest to readers of the English.

Summary

If we want to use Z to write an overall system specification, we need to integrate it into a rich set of documents written in natural language and domain-specific notations. These documents must be easy to write and read by non-mathematicians. I have described a tool and sketched a set of rules aimed at achieving this. There is much more to do, both in improving the tool and codifying the process for different domains.

References

1. Anthony Hall, *Using Formal Methods to Develop an ATC Information System*, IEEE Software, March 1996, pp 66-76.
2. Anthony Hall and Roderick Chapman, *Correctness by Construction: Developing a Commercial Secure System*, IEEE Software, Jan/Feb 2002, pp18 – 25.
3. Anthony Hall, *Seven Myths of Formal Methods*, IEEE Software, September 1990, pp 11-19.